An Adaptive Separation-Aware FTL for Improving the Efficiency of Garbage Collection in SSDs

Wei Xie, Yong Chen
Department of Computer Science
Texas Tech University
{wei.xie, yong.chen}@ttu.edu

Abstract— **Hot/cold data separation in flash-based solid state drives has been considered important to the overall performance due to the costly garbage collection overhead. This work proposes a method that accurately and naturally identifies and separates hot/cold data while only incurs minimal overhead. The proposed method only requires minimal on-device RAM space. Simulation results have shown that the proposed ASA-FTL reduce the GC overhead by up to 33% and improve the overall response time by 9% against the most advanced existing FTL in both real workloads and synthetic workloads.**

I. Introduction

In recent years, the emerged flash-memory based solid state drives (SSD) have fundamentally changed the landscape of storage systems and have rapidly replaced traditional hard disk drives (HDD) in many applications. Among different SSD devices, the NAND flash-based SSD gains the most popularity due to its high performance and relative low cost.

One critical characteristic of the flash-based SSD is that the data need to be erased before the flash memory can be written (programmed) again. The erasure operation is handled in *blocks*, and each block consists of multiple (usually 64 or 128) *pages*. The page is the smallest size that read and write operations are carried out. Due to the mismatch of the granularity of the erasure and the read/write operation, *out-of-place update* [1] scheme is adopted in most flash-based SSDs to allow acceptable write performance. This out-of-place write scheme makes the flash-based SSDs require a Flash Translation Layer (FTL) to manage the logical to physical address translation and a garbage collection when free flash space is exhausted.

Among different FTL schemes, the page-level FTL is considered performing best; but it requires considerable RAM space storing the mapping table, which is a critical limit for the inadequate on-device RAM space. The state-of-art Demand-based Selective Caching of Page-level Address Mapping scheme (DFTL) [1] leverages the page-level FTL scheme while significantly reduces the RAM usage by selectively caching the page-level mapping table. However, the garbage collection efficiency in the DFTL suffers since the mixture of hot/cold data in data blocks incur extra overhead in garbage collection. Separating hot/cold data into different physical space has the potential of reducing the garbage collection overhead.

This work aims at *increasing the efficiency the garbage collection of the page-level FTLs, by identifying and separating the hot/cold data online*. The proposed method leverages the page-level data access history as hotness information and a natural clustering algorithm to separate hot/cold data in the flash memory, meanwhile incurring minimal space and computational overhead with selective caching and random sampling of the hotness information.

Several recent studies have conducted hot/cold data identification and separation in a different context and for a different purpose. For instance, Chiang proposed an online hot/cold separation method that exploits the write frequency [2]. Multiple-pool scheme is introduced to support on-write data separation. The log-structured file system also uses the similar out-of-place write scheme and is considered suitable for flash-based SSDs [3]. The SSD friendly file system (SFS) exploits both the access recency and frequency for measuring hotness and adopts an algorithm that calculates the natural separation criteria based on the hotness information [3].

This work employs the hot/cold separation algorithm in the FTL layer similar to the one proposed in SFS. However, it differs from the SFS in following ways to leverage the restricted RAM space and computing power on board: (1) hotness information are randomly sampled for calculating the separation criteria; (2) hotness information are stored in flash memory and cached into RAM on-demand; and (3) the separation is fine-grained: in unit of pages.

II. ASA-FTL System Design

As demonstrated in Fig. 1, the proposed Adaptive

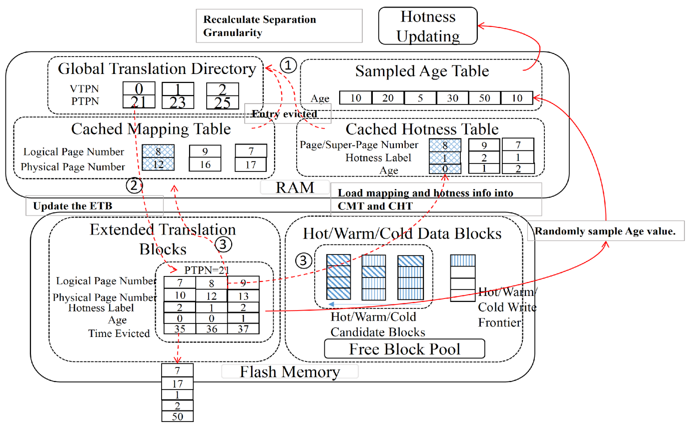

Fig. 1. System Architecture of ASA-FTL.

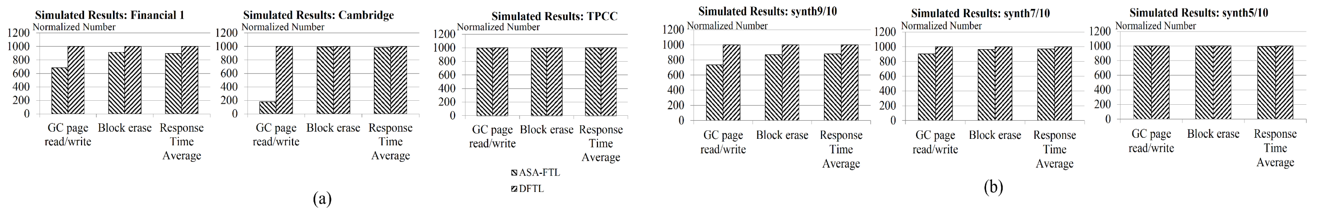

Fig.2. The simulation results of (a) real workloads and (b) synthetic workloads. All the traces are tested on ASA-FTL against DFTL. GC read/write count, block erasure count and average I/O response time are normalized and shown from left to right.

Seperation-Aware FTL (ASA-FTL) system consists of four major data structures maintained in RAM: (1) the Cached Mapping Table, where selective page mapping information are stored to accommodate incoming requests; (2) the Cached Hotness Table, where the access history (Age Since Last Update or Age) and hotness (Hotness Label 0, 1 and 2 corresponding to cold, warm, and hot) information of selective pages are stored so that the recency (Age) could be used at runtime to determine the hotness of data; (3) Global Translation Directory that maps the logical address to the physical address of the translation block in flash memory or the Extended Translation Blocks structure, which is used when a cache miss happens; and (4) Sample Age Table that stores a number of age values from random selected page, which are used as data sets for calculating the separation criteria.

The flash memory is separated into Data Blocks and Extended Translation Blocks (ETB), similar to the DFTL. The Data Blocks are separated into hot, warm and cold partitions to independently accept incoming write requests like the multiple-pool scheme [2]. The ETB permanently stores the mapping and hotness information of each page and loaded in CMT and CHT on-demand.

The Hotness Updating algorithm run in the background periodically samples the *Age* values from random pages into the SAT, and then leverages these values to calculate the separation criteria, using the K-means clustering algorithm. Thus when a write request comes, for example page 8 as shown in Fig.1, the ASA-FTL works as follows. (1) If the cache is full, it evicts one page (page 7) from both the CMT and CHT. (2) It looks up the GTD to retrieve the location of the entry in the ETB containing page 7 (which is in translation page 21) and the entry is updated. (3) It loads the information of page 8 into CMT and CHT. When loading the *Age value,* the time in ETB is added or *Age= 0(Age) + 50(Current Time) – 34(Time Evicted)*; (4) the *Age* of page 8 is compared to the criteria calculated and the destination partition is determined, then the write request is handled by the corresponding partition (hot, warm or cold) in the Data Blocks.

Only a small part of the Age values are sampled to save the on-device RAM space and the computing powered required by the K-means algorithm. The saved RAM space could be used for other purpose like data caching, read ahead etc, allowing more performance enhancement opportunities.

III. Preliminary Evaluation and Results

We have adopted the popular SSD simulator FlashSim in this study to implement and evaluate the proposed ASA-FTL. The ASA-FTL is implemented upon the DFTL and is evaluated against the state-of-the-art DFTL.

The ASA-FTL system was tested on 3 different real workloads and 3 synthetic workloads with varied skewness (workload with skewness means some data are more frequently accessed than others). The notation synth x/10 means x/10 of the write traffics go to (10-x)/x flash memory space. The rightmost chart in Fig.2 (b) shows the result of the trace with uniform traffic, which no difference is observed between ASA-FTL and DFTL.

As shown in Fig.2, the ASA-FTL significantly reduces the garbage collection overhead (GC page read/write) by 33% and overall performance (average response time) by 9% of the Financial1 and synth9/10 workload. This is due to their small request size and high degree of skewness. The effect of the ASA-FTL on the Cambridge and TPCC workload traces is much smaller due to their large request size and small skewness. Their garbage collection overhead is already close to optimal with the CAT policy, which is used in DFTL [4]. By comparing the simulation results of the synth9/10, synth7/10 and synth5/10, it can be seen that the benefit of the ASA-FTL decreases as the skewness of the workload reduces.

IV. Conclusion and Ongoing Work

Hot/cold identification and separation is an important method to make garbage collection more efficient and the SSD's overall performance improved. This paper proposes a FTL that effectively identifies and separates hot/cold data while only uses minimal on-device hardware resources. We are also working on unifying the garbage collection policy and wear leveling together to further improve the performance and expand the lifespan.

References

[1] Gupta, Aayush, Youngjae Kim, and Bhuvan Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. Vol. 44. No. 3. ACM, 2009.

[2] Chiang, M.-L., & Chang, R.-C. (1999). Cleaning policies in mobile computers using flash memory. Journal of Systems and Software. doi:10.1016/S0164-1212(99)00059-X

[3] Min, C., Kim, K., Cho, H., Lee, S.-W., & Eom, Y. I. (2012). SFS: random write considered harmful in solid state drives. *Proceedings of the 10th USENIX conference on File and Storage Technologies*. San Jose, CA: USENIX Association.

[4] Desnoyers, P. (2012). Analytic modeling of SSD write performance. In Proceedings of the 5th Annual International Systems and Storage Conference (p. 14). ACM